Machine Learning
Programming Assignment 4

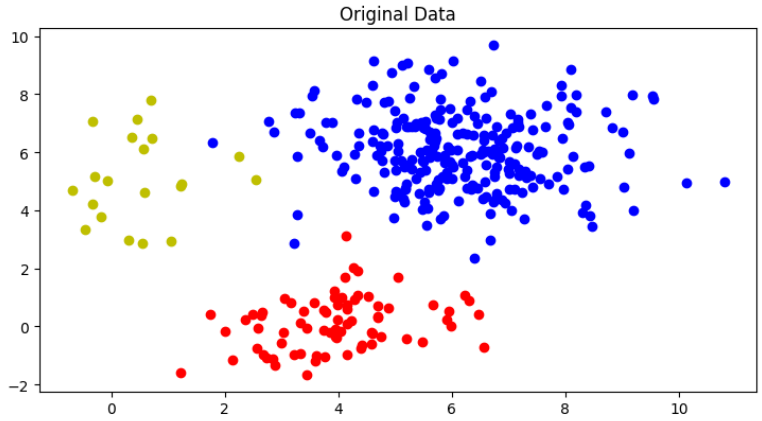
Vaibhav Jindal
111701029

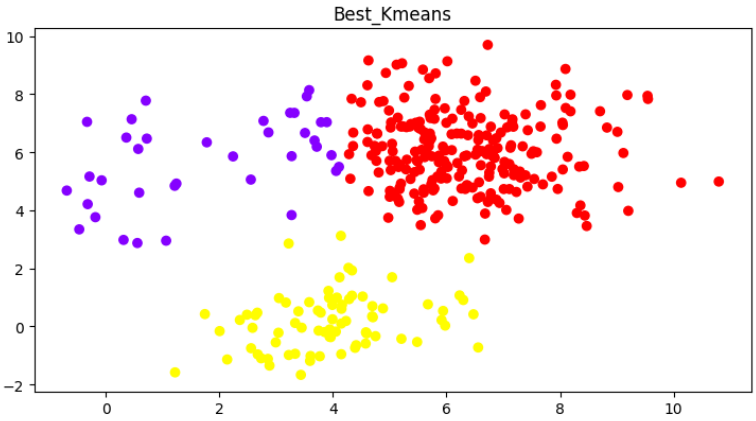
- Learned how to find good k (number of clusters) given the data points only. By adding a term dependent on k in cost function so that it doesn't shoot up k.
  - Example Cost = $\log(\Sigma_i^N || xi - \mu_{xi} ||^2_2 / m) + klogm/m$
  - μ_{xi} is the mean of cluster from which xi belongs
  - This cost function enusres that k doens't shoot up otherwise cost will be high
- Learned that k-means only gives spherical kind of clusters because cost is dependent of L_2 norm. Hence for elipsoidal clusters or other intersecting shapes we use Gaussian Mixture Model
- Learned about Mixing Coefficient (weightage given to each cluster), Covariance Matrices and Expectation Maximization method used to Learn GMM clusters.
- Learned about image compression using k-means.
- Finally learned how to implement all this using sklearn inbuilt library

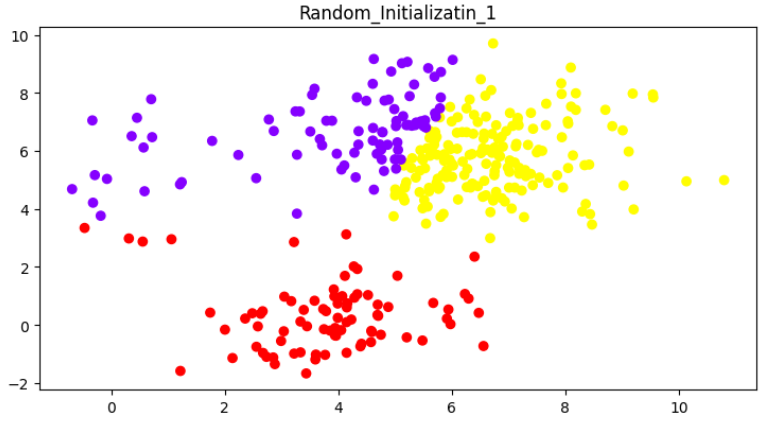
Task 1: Generate 3 clusters of different size and shape:

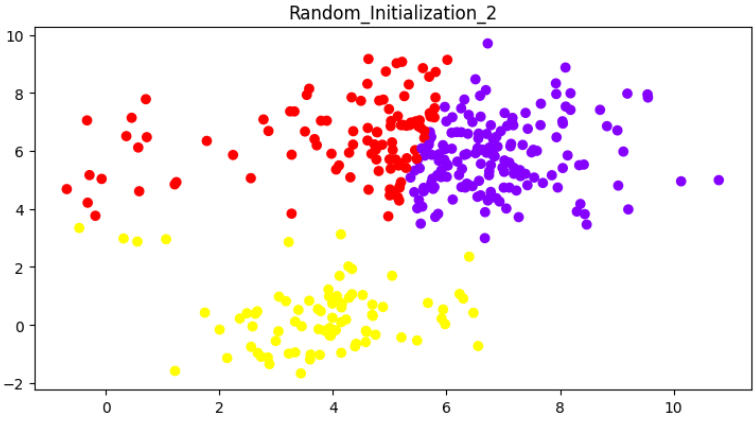
1. X_1 = np.random.multivariate_normal(mean=[4, 0], cov=[[1, 0], [0, 1]], size=75)
2. X_2 = np.random.multivariate_normal(mean=[6, 6], cov=[[2, 0], [0, 2]], size=250)
3. X_3 = np.random.multivariate_normal(mean=[1, 5], cov=[[1, 0], [0, 2]], size=20)

Observe the result of k-mean clustering for different initial position of centres









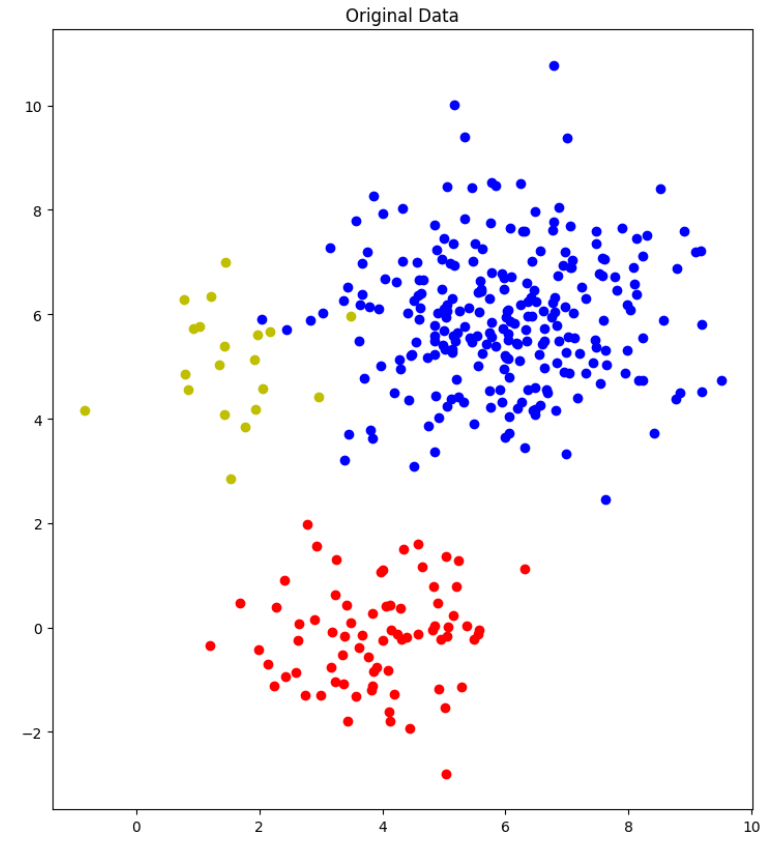
- It can be seen that it is very hard for k-means to detect the original cluster because the euclidean distance is almost the same from both the centres.
- Green dots in original cluster are coming in the other clusters.
- Also the number of data points of each cluster also matters. If there are more points of one cluster it is highly likely that bigger cluster will take over the smaller cluster points (or may even vanish the smaller one, example green one in original data are being taken by other clusters).
- Also k-means is sensitive to outliers as well as initial means.

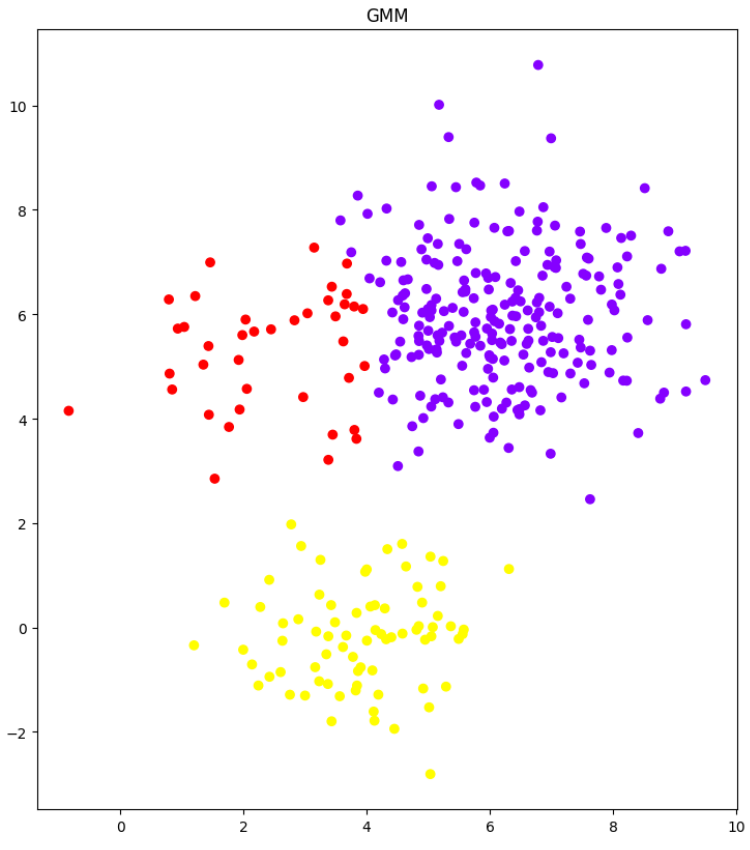
Initialization of centres with points from the input data points itlself.
Score on Best_Kmeans -1040.4173541130817

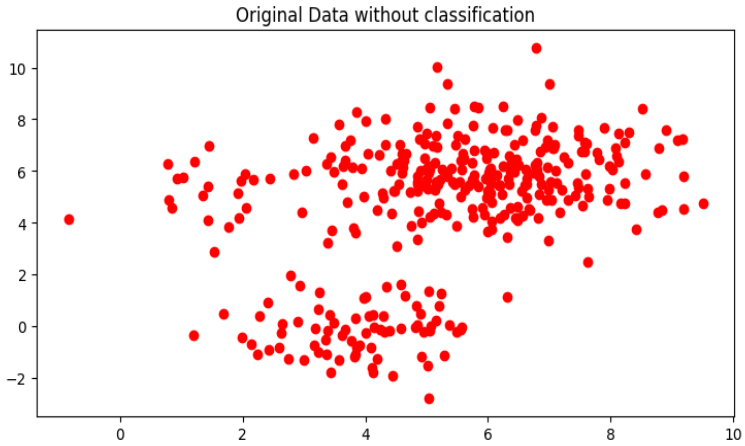
Random initialization of centres
Score on Random_Initializatin_1 -1044.9888545214149
Score on Random_Initialization_2 -1043.6599966843216

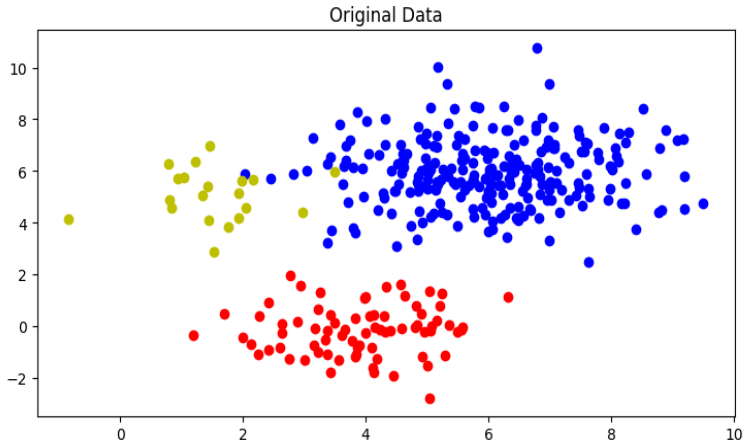
Score is better on within data points initialization of centres but not quite good.

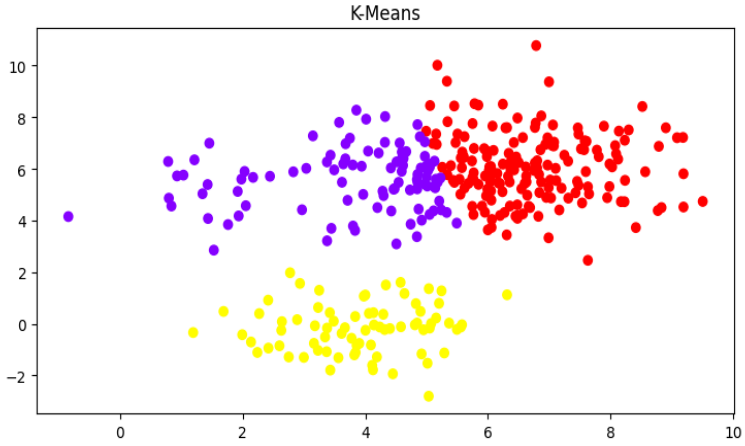
Task 2: Run GMM on same data and compare both methods

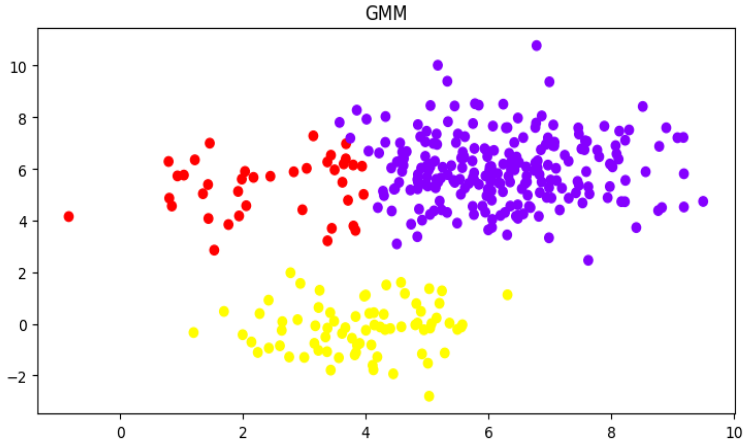












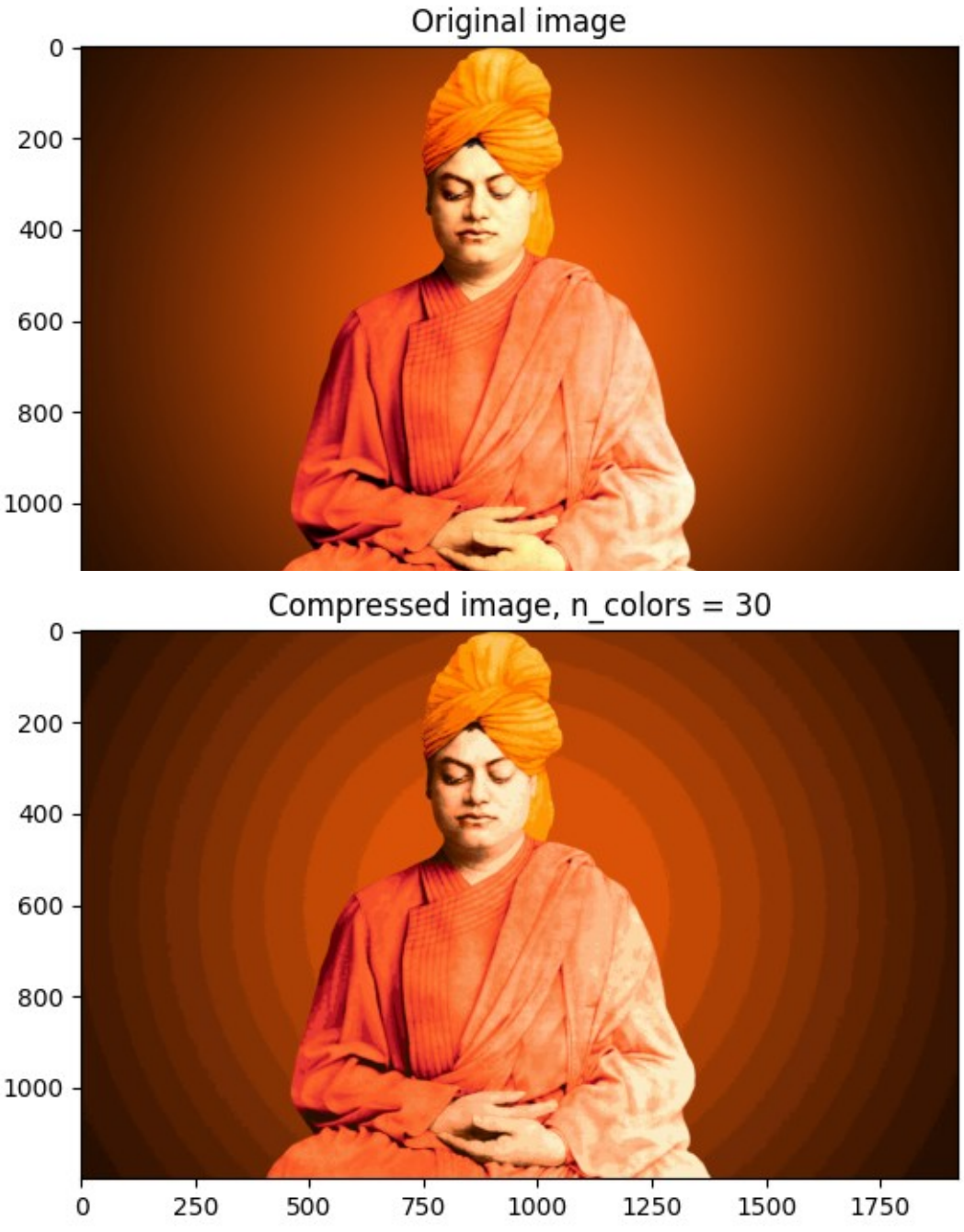
As it can be seen that GMM performs far better than k-means for the input data.
Reasons:

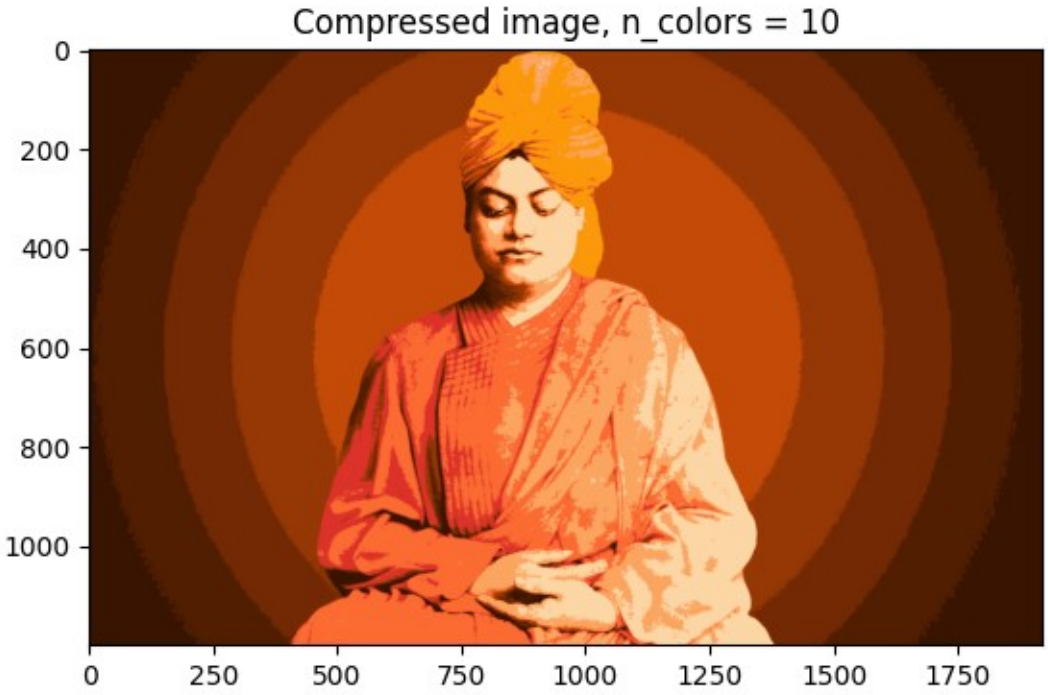
- GMM assumes that input data comes from a mixture gaussian distributions (which is indeed the case for us).
- GMM is more flexible to shape of cluster.
- GMM performs hard classification because it gives the probabilites that a given data point belongs to each of the possible clusters. K-means gives soft classification (gives only clusters).
- GMM can find intersecting clusters too.
- GMM has more parameters to learn.

Log-likelihood score on GMM -4.069738622448655 (Better than k-means)

Task 3:Implement k-means to compress an image. Take a high resolution RGB image. Therefore, for each pixel location we would have 3 8-bit integers that specify the red, green, and blue intensity values. Our goal is to reduce the number of colors to 30 and represent (compress) the photo using those 30 colors only. To pick which colors to use, we'll use k-means algorithm on the image and treat every pixel as a data point in 3-dimensional space which is the intensity of RGB. Will run kmeans to find 30 centroids of colors and finally we will represent the image using the 30 centroids for each pixel. Check if the actual size of the image changes.

K-means clustering will group similar colors together into 'k' clusters (here k=30) of different colors (RGB values). Each cluster centroid will represent the color vector. We need to only store the label for each pixel which tells the cluster to which this pixel belongs. Also we store the color vector of each cluster. (n_clusters,n_features)





- **With less number of clusters, encoding takes colors that have similar shades and makes them one single value.**
- We can see how the image is getting faded and less detailed. Quality is compromised.
- Image compression is lossy here.

Dimension of original image (1200, 1920, 3)
Dimension of first compressed image (1200, 1920, 3)
Dimenstion of second compressed image (1200, 1920, 3)
All have same dimensions but the required storage decreases with n_colors. Because the no of clusters decreases (No of features remain the same i.e. 3)

Conclusion:

1. K-means is a more general and easy to implement algorithm hence it has several uses.
2. GMM is good if we have proper k (no of clusters) and we know that data points come from a mixture of Gaussian Distribution
3. Image compression can also be done using other means which are loseless. Example: Huffman coding on the image.
4. It is hard to learn a proper k in general for clustering algorithms.